RESOLUTION OF THE UNIDIMENSIONAL CUTTING STOCK PROBLEM WITH USABLE LEFTOVER

Adriana Cristina Cherri
Marcos Nereu Arenales
Instituto de Ciências Matemáticas e de Computação – ICMC
Universidade de São Paulo – USP
Av. Trabalhador São-Carlense, 400 - Caixa Postal 668
13560-970 – São Carlos – SP
adriana@icmc.usp.br, arenales@icmc.usp.br

Horacio Hideki Yanasse
Laboratório Associado de Computação e Matemática Aplicada – LAC
Instituto Nacional de Pesquisas Espaciais – INPE
Av dos Astronautas, 1.758 – Jd. Granja
12227-010 – São José dos Campos – SP
horacio@lac.inpe.br

ABSTRACT

In this work we consider a one-dimentional cutting stock problem in which the non-used material in the cutting patterns may be used in the future, if large enough. This feature introduces difficulties in comparing solutions of the cutting problem, for example, up to what extent a minimum leftover solution is the most interesting one when the leftover may be used? Some desirable characteristics of good solutions are defined and classical heuristic methods are modified, so that cutting patterns with undesirable leftover (not large enough to be used, nor too small to be acceptable waste) are redesigned. The performance of the modified heuristics are observed by solving instances from the literature and practical instances.

Keywords: cutting stock problems, usable leftover.

1. Introduction

Cutting stock problems (CSP) consist in cutting large pieces (*objects*), available in stock, into a set of smaller pieces (*items*) in order to fulfill their requirements, optimizing a certain objective function, for instance, minimizing the total number of objects cut, minimize waste, minimize the cost of the objects cut etc. These problems are relevant in the production planning of many industries such as the paper, glass, furniture, metallurgy, plastics and textile industries.

Due to the diversity of situations where CSP arise, we are always faced with new constraints and/or objectives for which the available methods are of limited value. Hence, the use of simple heuristics has been observed in practice, many without any evaluation of their perfomance.

Although freqüently arising in practical situations, we could not find many articles in the literature that consider the situation where the leftover material may be used to cut future demands, if large enough. We call leftover any piece cut that is not a required item. To the best of our knowledge only Gradisar *et al.* (1997), Gradisar *et al.* (1999a), Gradisar *et al.* (1999b), Gradisar and Trkman (2005) and Abuabara (2006) consider this possibility. In 1997, Gradisar *et al.* proposed a heuristic (denoted by COLA) to optimize roll cutting in the textile industry with the objective of creating a cutting plan with reduced letfovers or to concentrate them in a single object. All objects have different lengths and they propose a bi-objective function that minimizes the number of unfulfilled item demands and the total loss (sum of the leftover smaller or equal to a pre-defined value). In 1999, Gradisar *et al*. proposed a modified COLA (denoted by CUT) and in 2005, Gradisar and Trkman developed an algorithm to find a solution to general unidimensional cutting stock problems with distinct objects, starting from the solution obtained by CUT and replanning patterns that do not satisfy some criteria. In 2006, Abuabara modified the mathematical model proposed by Gradisar *et al.* (1997), decreasing its size, that is, reducing the number of constraints and variables in the model.

In this work we present some characteristics of a desirable solution (we avoid "optimal solution" since a criterion to compare solutions is not defined) to the cutting stock problem with usable leftover (CSPUL). Modifications on classical heuristic methods to solve CSP are suggested aiming to find a solution that satisfies those characteristics.

This article is organized as follows. In Section 2, the CSPUL is defined. Some methods to solve it are presented in Sections 3 and 4. Computational tests are presented in Section 5 and conclusion remarks and future works are presented in Section 6.

2 Definition of the cutting stock problem with usable leftover

During the cutting process, unavoidable leftover occur that are often discarded. Some industries, however, have the possibility of using the leftover to cut future demanded items, as long as their sizes are sufficiently large. In this situation, the simple objective of minimizing the leftover may not be appropriate.

Many of the solution methods to solve cutting problems aim to minimize leftover (alternative objectives may be defined but low amount of leftover must also be pursued). Although a low amount of leftover is an objective to pursue, the possibility of using them introduces a new condition to evaluate a solution. In this new problem, planning cutting patterns that concentrate the leftover in fewer patterns seems to be a good alternative to pursue since it increases the chances that these leftovers will be sufficiently large to go back to stock to be used to cut future demanded items.

Hence, we present the unidimensional CSPUL as:

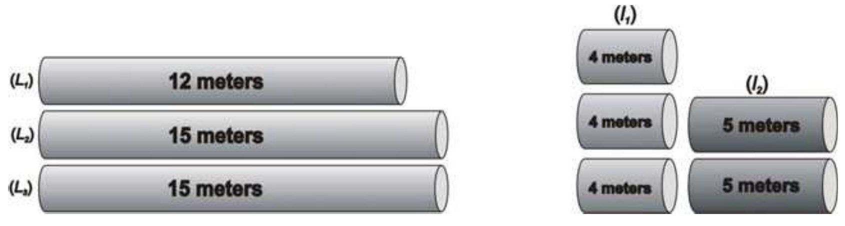
"A set of pieces (items) must be produced by cutting large units (objects) of standard sizes (objects bought from suppliers) or non standard (objects that are leftover of previous cuts). The demand of the items and the availability of the objects are given. Demand must be met by cutting the available objects such that the leftover are "small" (denoted by scrap) or "sufficiently large" (denoted by retail) to return to stock, but in a reduced number".

This high level definition aims to capture the main elements of the CSPUL but it lacks details that are going to be completed next.

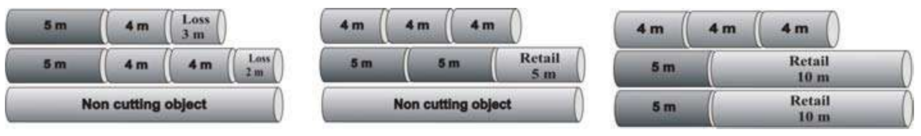
The "sufficiently large" length or, equivalently, the minimum acceptable length for retail is a choice of the decision maker. Some possible choices include the length of the shortest demanded item, the average lengths of the demanded items or the length of the longest demanded item. Gradisar *et al.* (1997), Gradisar *et al.* (1999a), Gradisar *et al.* (1999b) and Gradisar and Trkman (2005) considered a retail any leftover with length greater or equal to the shortest demanded item. This choice may not be interesting in cases where the portfolio of demanded items includes a small item that is not typical because it is likely that retails that are seldomly used will be stocked. On the other hand a particular portfolio may include only large items, and retails with sizes smaller than the smallest of the items are acceptable.

In the classical CSP we find objective functions like minimize the total waste, minimize the number of objects cut, minimize the costs, and so on. In the CSPUL our objective is to have little or no scraps (as in the classical problem) and/or a reduced number of retails. Therefore, two solutions with the same leftover may be different as illustrated in

Figure 1. In this example, a leftover piece of size larger or equal to 4 meters is considered retail.



(a) Objects in stock to be cut. **(b)** Required items.



(c) Solution 1. **(d)** Solution 2. **(e)** Solution 3.

Figure 1: A cutting stock problem data and alternative solutions

For the CSPUL, Solution 2 (Fig **1 - d**) is better than Solution 1 (Fig **1 - c**), since it concentrates the leftover of a size superior to 4 meters (a retail) in a single object (Solution 1 has 5 m of scrap while Solution 2 has zero scrap and a retail of 5 m). For the CSPUL we can say that Solution 1 is an *undesirable* solution compared to the *ideal* Solution 2. Another *undesirable* solution (compared with Solution 2) is Solution 3, given in Figure **1 - e**, for although it does not generate scraps, it generates a larger number of retails.

Due to the difficulty in defining a single objective function that differentiates such solutions we begin qualifying the solutions according to the following definition.

Definition 1: The solutions of a CSPUL are defined as:

- *Ideal solution: when a small number of objects have little scraps and none of the objects have not so little scraps. In case there are retails, they must be concentrated in a very small number of cut objects;*
- *Acceptable solution: when a small number of objects present not so little scraps and a small number of objects present retails;*
- *Undesirable solution: when several cut objects present not so little scraps or present several retails.*

Observe that an ideal solution is always acceptable but the reverse is not true.

This definition (that depends on quantifying terms like *small*, *very small* or *several objects, little scrap* or *not so little scrap* and *retail*), tries to incorporate general features of the solutions for the CSPUL. By not so little scrap we mean a leftover material that is larger than a little scrap but it is not big enough to be a retail.

The sizes of little scrap, not so little scrap or retail are defined by the user (decision maker). The decision maker can define these values by his/here experience. Also he/she may use parameters to define them, like:

θ: fraction that defines the largest size for a leftover material to be considered a *little scrap* for standard sized objects, that is, θL_k is the maximum size for a leftover material to be considered little scrap in a standard object of lenght L_k, $k = 1, ..., \bar{k}$, where $\bar{k}$ is the quantity of standard object types in stock;

- β: fraction that defines the largest size for a leftover material to be considered a *little scrap* for non standard sized objects, that is, βL_k is the maximum size for a leftover material to be considered little scrap in a non standard object of lenght L_k, $k = \bar{k} + 1, ..., K$ (the objects of type $\bar{k} + 1, ..., K$ are retails);
- δ: smallest size of a leftover to be considered a retail (for example, δ is the average length of the item types demanded). Any leftover larger or equal to δ is considered retail, independent of the object type.

Observe that with the parameters θ and β we make the scrap dependent on the object type. The additional parameter β allows the decision maker to define larger "little scraps" for non-standard objects, making them more prone to be used compared to the standard objects.

The quantities "*small*", "*very small*" and "*several*" in definition 1 are also defined by the user (decision maker). The decision maker can define them by his/her experience or he/she can use, for instance, two parameters ξ_1 and ξ_2, with $0 < \xi_1 < \xi_2 < 1$ and set:

- *Very small number of objects cut:* up to $\lceil \xi_1 \eta \rceil$ of the objects cut;
- *Small number of objects cut:* up to $\lceil \xi_2 \eta \rceil$ of the objects cut;
- *Several objects cut:* above $\lceil \xi_2 \eta \rceil$ of the objects cut;

where η is the total number of objects cut in the solution.

For simplicity, from now on, we use the term *acceptable leftover* when the leftover is a small scrap or a retail.

With the aim of generating an ideal solution, or at least an acceptable one, we introduce modifications in some well known heuristics of the literature to solve the unidimensional CSP so that solutions with several objects having not so small scraps are avoided. These are described in the next Sections 3 and 4.

3 Constructive Heuristics

One heuristic used in the solution of the CSP is the exhaustive repetition (Hinxman, 1980). This heuristic builds a "good" cutting pattern for each object type k, $k = 1, ..., K,$ select one of the cutting patterns generated (a selection criterion can be, e.g., minimum waste. This selected pattern is associated with an object type), use the cutting pattern chosen as much as possible, without exceeding the required demand of the items and the availability of the associated object and update the demand of the items and the stock of the objects. Two very well known procedures to generate "good" cutting patterns are FFD (*First Fit Decreasing*) and Greedy.

FFD Procedure

In the FFD procedure we initially cut the largest item as much as possible or until its demand is satisfied (the largest items are chosen first since they are, generally, more difficult to be combined). When it is not possible anymore to cut the largest item, the second largest item is considered and, so on, until we reach the smallest item. When no new item can be cut, the pattern is complete.

Greedy Procedure

In the greedy heuristic the cutting patterns are generated solving a sequence of knapsack problems of the form:

$$z(b) = \text{maximize } \ell_1 a_1 + \ell_2 a_2 + \ldots + \ell_m a_m \qquad (1)$$

$$\text{subject to:}$$

$$\ell_1 a_1 + \ell_2 a_2 + \ldots + \ell_m a_m \le b \qquad (2)$$

$$0 \le a_i \le r_i \text{ , } i = 1, \ldots, m \text{ and integer} \qquad (3)$$

where ℓ_i is the length of item i, $i = 1, ..., m$, and r_i is the residual demand of item i, $i = 1, ..., m$, updated in Step 4. Initially, $r_i = d_i$, $i = 1, ..., m$, the required number of items type i.

These two classical procedures have distinct phylosophies. In the FFD procedure, there is an excessive concern in producing the largest items as earlier as possible since they are harder to combine, while in the Greedy procedure the best possible cutting patterns are generated first, with no concern about the quality of the future cutting patterns. Next we modify these two procedures to deal with the case of usable leftover.

3.1 FFD$_L$ procedure

We modify the FFD procedure aiming to avoid not so small scraps, that is, trying to obtain at least, acceptable solutions.

The FFD$_L$ procedure applies FFD to generate a pattern and, just after the generation, the leftover is analysed. If it is an acceptable leftover (a small scrap or a retail) the pattern is accepted. Otherwise, we take out an item (the largest one) of the pattern. With the resulting empty space we solve the knapsack problem (1)-(3), with the capacity b equal to the leftover material of the generated pattern plus the size of the item withdrawn. After solving the knapsack problem, the resulting leftover is analysed and if it is not acceptable, an additional item (second largest) of the original pattern is withdrawn. Again, with the space generated a new knapsack problem (1)-(3) is solved. In case we have withdrawn an item of each size among all items in the pattern, we withdraw again another piece of the largest item. This procedure is repeated until the leftover obtained is acceptable or the initial cutting pattern becomes null. In this latter case, the cutting pattern is the solution of knapsack problem (1)-(3). For more details, see Cherri *et al.* (2007).

3.2 Greedy$_L$ procedure

The Greed$_L$ procedure consists in applying the Greedy procedure to obtain a cutting pattern and observe the leftover. If acceptable (small scrap or retail), the pattern is accepted, otherwise the largest item in the pattern is withdrawn and the leftover is analysed again. If the pattern is still unacceptable the second largest item is withdrawn. This process is repeated until we have an acceptable pattern or a null pattern. If the pattern is null, we choose among the original patterns, the one that presents the smallest leftover (this is not a typical situation but it can occur, for instance, when the stock is formed only by retails). For more details, see Cherri *et al.* (2007).

4 Residual Heuristics

Residual heuristics find an integer solution for the unidimensional CSP from the linear relaxation of the Integer Programming problem

$$\text{Minimize } f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \qquad (4)$$

$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{d} \qquad (5)$$

$$\mathbf{E}\mathbf{x} \le \mathbf{e} \qquad (6)$$

$$\mathbf{x} \ge \mathbf{0} \text{ and integer} \qquad (7)$$

each column of the matrix **A** is associated with a cutting pattern, and **E** is a matrix of 0's and 1's with stock constraints. For detail, see Poldi and Arenales (2005).

The integrality condition on the variables x_{jk} complicates the solution of the problem as m increases (it is already difficult when m is of the order of a few dozens). Gilmore e Gomory (1961) proposed to solve the problem using *column generation* after relaxing the integrality constraints on the decision variables x_{jk}. From the optimal solution of the relaxed problem, that usually is not integer, we determine an integer solution for the original CSP. This integer solution can be determined by heuristics that have been developed by several researchers, like Wäscher and Gau (1996), Poldi and Arenales (2005), Stadtler (1990), among others. Some of these heuristics are presented in subsections 4.1 and 4.2.

Definition 2: Let **x** be an optimal fractional solution for the linear realxation problem (4)-(7) and let **y** be a vector of non-negative integer numbers, close in some sense to **x**, such that:

$$\mathbf{Ay} \leq \mathbf{d} \tag{8}$$

$$\mathbf{Ey} \leq \mathbf{e} \tag{9}$$

y, obtained from **x**, is called an approximate integer solution of **x**.integer solution **y** of **x** is by a simple truncation:

$$\mathbf{y} = (\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \ldots, \lfloor x_n \rfloor) \tag{10}$$

that satisfies (8)–(9) since all coefficients of **A** and **E** are non-negative, **x** satisfies **Ax = d** and **Ex** ≤ **e**.

Definition 3: (Residual Problem): Let **y** be an approximate integer solution of **x**, **r = d – Ay** the residual demand and **s = e – Ey** the residual stock of the available objects. The residual problem is defined as (4) – (7) with **d** = **r** and **e** = **s**.

In residual heuristics we solve a linear relaxation of problem (4)-(7) and we obtain an approximate integer solution. We then solve a linear relaxation of the residual problem (definition 3) and we obtain an approximate integer solution and, so on successively, until the residual demand is null or the approximate integer solution is null. In the latter case, we apply some method (heuristic or exact) to solve the residual problem with just a few items. Next, we present a general structure of these heuristics.

Residual Algorithm (from Poldi and Arenales, 2005)

```
Step 1: {Start}
    Do $\ell = 0$, $\mathbf{r}^0 = \mathbf{d}$, $\mathbf{s}^0 = \mathbf{e}$;

Step 2: {Determining the continuous optimal solution}
    Solve the residual problem with $\mathbf{r} = \mathbf{r}^\ell$ and $\mathbf{s} = \mathbf{s}^\ell$;
    Let $\mathbf{x}^\ell$ be the continuous solution (column generation is used);
    If $\mathbf{x}^\ell$ is integer, then STOP.

Step 3: {Determining an approximate integer solution}
    Determine $\mathbf{y}^\ell$, the approximate integer solution of $\mathbf{x}^\ell$.
    If $\mathbf{y}^\ell$ is a null vector, then go to the Step 5.

Step 4: {Update}
    Determine the new residual demand
    $\mathbf{r}^{\ell+1} = \mathbf{r}^\ell - \mathbf{A}\,\mathbf{y}^\ell$;
    $\mathbf{s}^{\ell+1} = \mathbf{s}^\ell - \mathbf{E}\,\mathbf{y}^\ell$;
    $\ell = \ell + 1$.
    Go to Step 2.
```

Step 5:

Solve the final residual problem with a few items by some method, heuristic or exact.

4.1 Residual Heuristic by Greedy Rounding (RGR)

Poldi and Arenales (2005) developed a greedy rounding procedure to obtain an approximate integer solution of a continuous solution **x** in Step 3 of the residual algorithm.

In Poldi and Arenales's greedy rounding procedure, Step 3 of the residual algorithm is divided in two parts: *Pre-processing* Step and *Rounding* Step. For these heuristics, refered to as RGR heuristics, the Step 5 of the residual heuristic never occurs for all demand is satisfied since at least for one pattern its frequency will be rounded up to its nearest integer and, hence, $\mathbf{y}^{\ell}$ will never be null in Step 3.

In the *Pre-processing* Step we order the cutting patterns of the continuous solution $\mathbf{x}^{\ell}$ (obtained in Step 2 of the residual algorithm) according to RGR 1, RGR 2 and RGR 3, described in Poldi and Arenales (2005).

In the *Rounding* Step, we start with the first cutting pattern, following one of the previous orderings, and we round up its frequency, that is, $y_{1k_1} = \lceil x_{1k_1} \rceil$. The other frequencies are set to 0, that is, $y_{jkj} = 0$, j = 2, ..., *T*. Conditions (8) and (9) are tested and, in case of violation, y_{1k_1} is reduced successively by a unit, that is, $y_{1k_1} = y_{1k_1} - 1$, unitl we get an approximate integer solution. The value of y_{1k_1} is fixed and we repeat the procedure with the second pattern. We determine a new approximate integer solution y_{1k_1}, y_{2k_2} and $y_{jkj} = 0$, j = 3, ..., *T*. We repeat the procedure for all cutting patterns.

To solve the CSPUL, we adapted this greedy rounding to obtain heuristics RGR$_L$ – versions 1, 2 and 3.

4.2 RGR Heuristics for the CSPUL

In the RGR$_L$ heuristics we use a bound for the maximum acceptable fraction for a scrap defined from the approximate integer solution obtained by using one of the versions of the RGR heuristic (in the constructive heuristics, this bound is previously defined by the user, for standard and non-standard objects).

The RGR$_L$ heuristics consist of determining an approximate integer solution by using the greedy rounding procedure according to RGR (versions 1, 2 or 3). After, we analyze the leftover material of all cutting patterns generated. If the leftover is *acceptable*, *i.e*., if it is less than the limit previously calculated, then the pattern *i* of object type k_i is accepted and stored, otherwise, the pattern *i* is rejected and demand of the items in pattern *i* and inventory of the object type k_i are updated. After analyzing all the cutting patterns generated, we apply the FFD$_L$ procedure to solve the residual problem formed by the items of the rejected patterns and the remaining objects.

Other procedures to solve the residual problem in the heuristics type RGR$_L$ can be used. We suggest the FFD$_L$ procedure because this heuristic generates less scrap when compared with the FFD and Greedy heuristics, and a smaller quantity of objects cut with retail compared to Greed$_L$ heuristic.

5 Computational Tests

Computational tests are presented by solving instances from Trkman (2005) and practical instances from Abuabara (2006). For these tests, we consider retail any material left with length larger or equal to the average of the lengths of the required items.

To classify the solutions (*ideal*, *acceptable*, or *undesirable*) obtained according to definition 1, we use the values $\xi_1 = 0.03$ and $\xi_2 = 0.1$, except for instances 4 and 5, that present small demand.

All the heuristics were implemented in DELPHI 6. The experiments were executed in a Pentium IV (3 GHz, 2 GB RAM) microcomputer. The implicit enumeration method described in Gilmore and Gomory (1961) was used to solve the knapsack problems that arise in the heuristics and in the column generation.

5.1 Results using Instances of the Literature

In this section we present numerical instances from Trkman (2005) and practical instances from Abuabara (2006).

In the instances of Trkman (2005) we have several types of objects in stock but only one of each type, that is, $e_k = 1$ for all k. This is a very special situation where the variables x_{jk} are binary. Due to the special characteristic of these instances, we treated the objects as non-standards and we set only the parameter β to 0.005.

For these instances we have information about the solution given by the CUT algorithm, therefore, we used the same criteria adopted by Gradisar *et al.* (1997) for *scrap* and *retail*, that is, all the material left after cutting an object that is greater or equal to the smallest item demanded is considered *retail*, otherwise it is *scrap*.

In the following tables we classify the solutions according to definition 1 and we use **ID** to denote an *ideal solution*, **AC** to denote an *acceptable solution*, and **UND** to denote an *undesirable solution*. Also we use Obj.Cut. to denote the quantity of objects cut, Tot.Length to denote the total length of the objects in stock cut, Total Loss to denote the total length of scraps, Total Ret. to denote the total length of the retails, OSScrap to denote the number of objects cut with small scrap, ONSScrap to denote de number of objects cut with not so small scrap, and ORetail to denote the number of objects cut with retail.

Instance 1. $K = 20$ types of objects with lengths between 2,200 and 6,000 cm; availability of one unit of each type of object and $m = 5$ items demanded according to Table 1.

Table 1 - **Data of instance 1 – Items**

Item	Length (cm)	Demand
1	235	4
2	200	51
3	347	42
4	471	16
5	274	37

From Table 1, $\delta = 200$ cm since this is the length of the smallest item. In Table 2 we present the computational test results obtained.

Table 2 – **Solution of Instance 1**

		Constructive				**Residual**					
	CUT	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	11	11	13	12	12	10	10	10	10	11	10
Tot.Length	44136	44027	48506	44715	46104	44079	45245	44079	45245	47141	46507

Total Loss	5	639	8	39	1	0	0	0	0	19	0
Total Ret.	743	0	5110	1288	2715	691	1857	691	1857	3734	3119
OSScrap	3	1	3	1	1	0	0	0	0	0	0
ONSScrap	0	10	0	3	0	0	0	0	0	1	0
ORetail	1	0	7	1	4	1	1	1	1	2	2
Solution	AC	UND	UND	UND	UND	**ID**	**ID**	**ID**	**ID**	AC	AC

From Table 2 we observe that the heuristics RGRL – versions 1, 2 use less objects than the CUT algorithm but the total length of the objects is longer, that is, they use longer objects compared to CUT. This was expected since CUT gives priority to smaller objects first. We also notice that similar to algorithm CUT, heuristics RGRL – versions 1 e 2 concentrate the leftover in a single object. With respect to losses, none of these two heuristics generate losses performing better than algorithm CUT with respect to this criterion. Note that, in general, the derived heuristics with usable leftover perform better compared to their original version by definition 1. For this instance, the classification of the solutions obtained by each algorithm is given in the last row of the Table, using definition 1 and the parameters defined previously.

Instance 2: $K = 90$ types of objects with lengths between 3,000 and 9,000 cm; availability of one unit of each type of object and $m = 15$ item types demanded according to Table 3.

Table 3 – **Data of instance 3 – Items**

Item	**Length (cm)**	**Demand**
1	569	34
2	718	26
3	520	25
4	540	12
5	492	30
6	547	2
7	632	6
8	430	36
9	750	7
10	387	20
11	804	3
12	389	32
13	835	18
14	684	39
15	687	10

For this instance, δ is equal to 387 cm.

Table 4 - Solution of Instance 3

		Constructive				Residual					
	CUT	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	27	27	29	30	30	22	22	22	22	24	22
Tot.Length	170504	172055	170343	175473	173814	170621	172114	170621	172114	175742	170989
Total Loss	2	2009	14	224	0	0	0	0	0	0	0
Total Ret.	1456	1000	1283	6203	4768	1575	3098	1575	3098	6696	1943
OSScrap	2	7	9	0	0	0	0	0	0	0	0
ONSScrap	0	17	0	4	0	0	0	0	0	0	0
ORetail	1	1	1	2	6	2	1	2	1	3	1
Solution	**ID**	UND	AC	UND	UND	AC	**ID**	AC	**ID**	AC	**ID**

For this larger instance, we observe from Table 4 that the solutions presented by the residual heuristics RGR$_L$ – verions 1, 2 and 3 and by the CUT algorithm were considered *ideal*, according to definition 1. Also, if we compare the solutions obtained with the modified heuristics compared with the ones obtained with the original procedures, we clearly observe improvements in quality, following definiton 1.

We presented here the computational test results of 2 instances only, although we received and tested a total of eight instances. In all the 8 instances we observed a similar behaviour, i.e., the performance of the modified heuristics were better or equivalent to the performance of algorithm CUT, according to definiton 1. The results of the other instances can be seen in Cherri (2006).

The next set of instances is from Abuabara (2006), whose work is based on the models proposed by Gradisar *et al.* [3]. The instances are from the portfolio of demands of a small Brazilian agricultural airplane industry that cuts methalic tubes to build its airplanes whose structure are formed with lattice porticoes.

The instances presented, which characterize small industries, are not classified according to definition 1 since the values $\xi_1 = 0.03$ and $\xi_2 = 0.1$ that we planned to use are not appropriate because the total number of objects cut is small (small demand) therefore yielding a very small bound for the quantity of objects with small scraps, not so small scraps and retails. The choice of the best solution according to definition 1 can be made after the quantities "*small*", "*very small*" and "*several*" are defined by the decision maker.

In the following examples, we have a single type of object in stock, that is, $e_k = 1$, $k = 1$.

Since the stock has standard objects only we define solely the parameter θ and we use $\theta = 0.005$. The minimum size of the retails is δ = min, i = 1, ..., *m*}.

Instance 3: The length of the objects in stock is 3,000 cm and there are 10 of them. $m = 5$ item types demanded according to Table 5.

Table 5 – **Data of instance 4 – Items**

Item	Length (cm)	Demand
1	250	2
2	273	2
3	285	4
4	525	4
5	1380	4

Any leftover material on a cutting pattern larger than or equal to 250 cm is considered a retail.

Table 6 – **Solution to Instance 4**

	Constructive				Residual					
	FFD	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	4	5	4	5	4	5	4	5	4	5
Tot.Length	12000	15000	12000	15000	12000	15000	12000	15000	12000	15000
Total Loss	525	0	240	0	240	0	240	0	244	4
Total Ret.	1669	5194	1954	5194	1954	5194	1954	5194	1950	5190
OSScrap	0	0	0	0	0	0	0	0	1	1
ONSScrap	3	0	1	0	1	0	1	0	1	0
ORetail	1	3	1	3	1	3	1	3	1	3

From Table 6 we observe that the RGR$_L$ algorithms – versions 1, 2, do not generate scraps but they present large quantities of retails when compared to the original heuristics. This occurs since the usable leftover heuristics tend to eliminate the not so small scraps.

In this instance and the next ones, we are faced with solutions where we have loss of material but a single retail or no loss of material but a larger number of retails. This is a typical situation that we face when we solve this problem; we rarely obtain a solution that is good considering all the criteria. So, the choice of the best solution is of the decision maker since he/she knows better the reality of the firm.

Instance 5: The length of the objects in stock is 6,000 cm and there are 10 of them. m = 4 item types demanded according to Table 7.

Table 7 - **Data of instance 5 – Items**

Item	Length (cm)	Demand
1	370	5
2	905	5
3	910	5
4	930	5

Any leftover material on a cutting pattern larger than or equal to 370 cm is considered retail.

Table 8 – **Solution to Instance 5**

	Constructive				Residual					
	FFD	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	3	4	3	3	3	3	3	3	3	3
Tot.Length	18000	24000	18000	18000	18000	18000	18000	18000	18000	18000
Total Loss	250	0	250	0	515	150	515	150	515	150
Total Ret.	2175	8425	2175	2425	1910	2275	1910	2275	1910	2275
OSScrap	0	0	0	0	0	0	0	0	0	0
ONSScrap	2	0	2	0	2	1	2	1	2	1
ORetail	1	4	1	3	1	2	1	2	1	2

As we can observe, the original heuristics generate only a single retail to stock but all of them present larger losses compared to the losses of the modified heuristics. In the RGR$_L$ heuristics– versions 1, 2 and 3, it was not possible to eliminate all the not so small scraps, but the quantity of retails is smaller compared to the other modified heuristics. The choice of the best solution is not trivial since it involves the simultaneous analysis of several features. The option of generating only a single retail leads to significantly larger losses while the solutions with no loss have a significant number of retails. Again, the instance shows the conflicting nature of the objectives and the decision maker must make his/her choice.

For these instances, solutions with the mathematical models of Gradisar et al. (1997) or Abuabara (2006) can be obtained. In these models constraints on the number of retails are imposed. The solutions obtained with these models are, in general, similar to the solutions obtained using the modified heuristics. When we consider a great variety of object types in stock with large availability as well as a large variety of item types with large demands, the mathematical models present a large number of variables and advanced solvers like CPLEX, often are not able to produce a solution.

To evaluate the heuristics described in sections 3 and 4, a test generator was elaborated and 16 classes of instances were considered. For each class, 20 instances were randomly generated. For these randomly generated classes, we also present the results obtained using the COLA algorithm developed by Gradisar *et al.* (1997). Algorithm COLA minimizes the loss of material and/or tries to concentrate them in a single object so that they become retail. After analyzing tests, we verify that the RGR$_L$ type heuristics present the best solutions, *i.e*., they were classified as *ideal*. Detail of the instances and solutions can be found in Cherri *et al.* (2007).

6 Conclusions

In this article we considered the cutting stock problem with usable leftover, that is, if the resulting leftover material of a cut object is large enough it can be used again to cut future demanded items. To deal with this problem, we modified some heuristics of the literature that minimizes the trim loss and we included the possibility of retails (large leftover) that are not computed as losses. A set of desirable characteristics were used to define solutions as *ideal*, *acceptable* and *undesirable*. Still, there exist difficulties in pointing out which method performed better for the solutions present important and conflicting characteristics, like retail, small scrap, not so small scrap, together with their distribution in the cut objects. Other characteristics like the total length of the losses generated, the quantity of new retails generated, can also be included.

The use of mathematical models like those proposed by Gradisar *et al.* (1997, 1999a, 1999b, 2005) and Abuabara (2006) are practically suitable for solving problems with a small quantity of objects and items of moderate sizes but they are computational time consuming for instances with large quantities of objects and items with large demands. The modified heuristics can handle these instances without much effort.

Acknowledgments

The authors are indebed to Dr. Kelly Cristina Poldi for her help on implementations. This work was partially financed by FAPESP and CNPq.

References

ABUABARA, A., (2006), Otimização no corte de tubos estruturais: aplicação na indústria aeronáutica agrícola. MS Dissertation, DEP - UFSCar, São Carlos, SP, Brazil.

CHERRI, A. C., (2006), O problema de corte de estoque com reaproveitamento da sobras de material. MS Dissertation, ICMC - USP, São Carlos, SP, Brazil.

CHERRI, A. C., ARENALES, M., YANASSE, H. (2007), The unidimensional cutting stock problem with usable leftover – a heuristic approach. Notas do ICMC - Série Computação, 91, ICMC – USP, São Carlos, SP, Brazil.

GRADISAR, M., JESENKO, J., RESINOVIC, C., (1997), Optimization of roll cutting in clothing industry. Computers & Operational Research, 10: 945-953.

GRADISAR, M., KLJAJIC, M., RESINOVIC, C., JESENKO, J., (1999a), A sequential heuristic procedure for one-dimentional cutting. European Journal of Operational Research, 114: 557-568.

GRADISAR, M., RESINOVIC, C.,KLJAJIC, M., (1999b), A hybrid approach for optimization of one-dimentional cutting. European Journal of Operational Research, 119: 719-728.

GRADISAR, M., TRKMAN, P., (2005), A combined approach to the solution to the general one-dimentional cutting stock problem. Computers and Operations Research, 32: 1793-1807.

GILMORE, P. C., GOMORY, R. E., (1961), A linear programming approach to the cutting stock problem. Operations Research, 9: 848-859.

HINXMAN, A., (1980), The trim-loss and assortment problems: a survey. European Journal of Operational Research, 5: 8-18.

TRKMAN, P., (2005), Private Communication (09/11/2005).

POLDI, K. C., ARENALES, M. N., (2005), Dealing with small demand in integer cutting stock problems with limited different stock lengths. Notas do ICMC - Série Computação, 85, ICMC – USP, São Carlos, SP, Brazil.

STADTLER, H., (1990), A one-dimensional cutting stock problem in the Aluminium Industry and its solution. European Journal of Operational Research, 44: 209-223.

WÄSCHER, G., GAU, T., (1996), Heuristics for the integer one-dimensional cutting stock problem: a computational study. OR Spektrum, 18: 131-144.